Asignatura: Fundamentos de algoritmia
Evaluación continua. Cuestionario.
Profesor: Isabel Pita.

Nombre del alumno:

1. Dada una serie de algoritmos cuyos ordenes de complejidad se dan a continuación, indica el mayor tamaño de entrada con el que se ejecutarían en un tiempo razonable. Lo relevante es el orden de magnitud: 10, 100, 1.000 etc.

Complejidad	Tamaño entrada
$\mathcal{O}(n^2)$	1.000-10.000
$\mathcal{O}(n)$	100.000-1.000.000
$\mathcal{O}(2^n)$	50
$\mathcal{O}(n^n)$	10
$\mathcal{O}(\log n)$	$10^6 - 10^7$
$\mathcal{O}(n \log n)$	10.000-100.000
$\mathcal{O}(1)$	Se ejecuta en el mismo tiempo para cualquier entrada

2. Escribe tres propiedades de los órdenes de complejidad y explícalas.

   Respuesta:

   a) $\mathcal{O}(a.f(n)) \equiv \mathcal{O}(f(n))$. Las constantes multiplicativas no afectan al orden de complejidad.

   b) $\mathcal{O}(f(n) + g(n)) \equiv \mathcal{O}(max(f(n), g(n)))$. El orden de complejidad de la suma de dos funciones es el máximo de ellas.

   c) $\mathcal{O}(log_a n) \equiv \mathcal{O}(log_b n)$. Todos los logaritmos pertenecen al mismo orden independientemente de su base.

3. Dado el siguiente programa que elimina los valores negativos de un vector, indica su orden de complejidad justificando tu respuesta. No olvides indicar la magnitud respecto a la cual indicas el coste.

```
void f (std::vector<int> & v) {
  int i = 0;
  while ( i < v.size())
    if (v[i] < 0) {
        for (int j = i; j < v.size()-1; ++j) v[j] = v[j+1];
        v.pop_back();
    }
    else ++i;
}
```

   Respuesta:

   El algoritmo es cuadrático en el número de elementos del vector.

   La función de coste es $\sum_{i=0}^{v.size()}(n-i) = n^2 - \sum_{i=0}^{v.size()} i = n^2 - \frac{n^2}{2} = \frac{n^2}{2} \in \mathcal{O}(n^2)$.

4. Dado el siguiente programa que calcula la primera potencia de dos mayor que un valor, indica su orden de complejidad justificando tu respuesta. No olvides indicar la magnitud respecto a la cual indicas el coste.

```
int f (int numero) {
    int pot = 2;
    while (pot <= numero) pot *= 2;
    return pot;
}
```

Respuesta:

El algoritmo tiene complejidad logarítmica en el número de entrada $\mathcal{O}(\log\ numero)$, siendo $numero$ el valor de entrada a la función.

El coste de la función es el coste del bucle `while` ya que la primera y la última instrucción tienen coste constante. El bucle da $\log\ numero$ vueltas, ya que en cada vuelta la variable `pot` se multiplica por dos, por lo que alcanza el valor de número en $\log\ numero$ vueltas. El coste de cada vuelta es constante ya que solo se realiza una multiplicación y una asignación.

5. Tenemos dos algoritmos que resuelven el mismo problema y cuyas funciones de coste son: n^2 y $1000n$

   Indica que que algoritmo ejecutarías y en que condiciones. Justifica tu respuesta.

   Respuesta:

   Se ejecutaría el algoritmo de coste cuadrático para valores de entrada menores que 1000 y el algoritmo lineal para valores mayores que mil. Esto se debe a que para valores de n menores que 1000 $n^2 < 1000n$ y para valores mayores que 1000 se tiene $n^2 > 1000n$.

6. Dada la siguiente expresión

$$\#k : 0 < k < v.size() - 1 : v[k-1] < v[k] < v[k+1].$$

   Indica que valor toma sobre el vector 4 3 5 7 9 5 5 6 7 2. Justifica tu respuesta.

   Respuesta:

   Debemos contar el número de valores que son mayores que el valor anterior y menores que el valor siguiente. La expresión toma el valor tres sobre el vector dado. Los valores que cumplen la propiedad son el primer 5, el primer 7 y el 6.

7. Escribe un predicado $diferentes(v)$ que exprese que todas las componentes de un vector v son diferentes.

   Respuesta:

   Existen muchas formas de expresarlo, algunas son:

   - $diferentes(v) \equiv \forall k1, k2 : 0 \leq k1 < k2 < v.size() : v[k1] \neq v[k2]$.
   - $diferentes(v) \equiv \forall k1, k2 : 0 \leq k1 < v.size() \wedge 0 <= k2 < v.size() \wedge k1 \neq k2 : v[k1] \neq v[k2])$.
   - $diferentes(v) \equiv \forall k : 0 <= k < v.size() : ((\#x : 0 <= x < v.size() : v[k] == v[x]) == 1)$.

8. Especifica un algoritmo que reciba un vector de entrada cuyas componentes son todas diferentes y compruebe si están ordenadas en orden estrictamente decreciente. Utiliza el predicado definido anteriormente.

   Respuesta:

   Nos piden una especificación, por lo tanto debemos dar una precondición P, una cabecera de la función para indicar cuales son los parámetros de entrada y los valores de salida y una postcondición Q.

   P: $\{v.size \geq 0 \wedge diferentes(v)\}$
   ordenado(vector `<int>` v) dev `bool` s
   Q:$\{s \equiv \forall k : 0 \leq k < v.size - 1 : v[k] < v[k+1]\}$

9. Escribe una expresión $maxPares(v)$ que calcule el máximo de los valores en las componentes pares de un vector.

   Respuesta:

   Una expresión toma un valor entero. Se construyen con las expresiones $\sum$, $\prod$, máx, mín o #.

   $maxPares(v) \equiv \texttt{max}\ k : 0 \leq k < v.size \wedge k\,\%2 == 0 : v[k]$

10. Utiliza la expresión anterior para especificar un algoritmo que dado un vector de al menos dos elementos compruebe que el máximo de las componentes se encuentra posiciones pares y no impares.

    Respuesta:

    De nuevo nos piden una especificación. Se debe dar una precondición, la cabecera de la función y una postcondición.

    P:$\{v.size > 1\}$
    maximoPar(vector `<int>` v) dev bool s
    Q:$\{s \equiv \forall k : 0 \leq k < v.size \wedge k\,\%2 == 1 : maxPares(v) > v[k]\}$